**周报**

本周学习了《算法设计与分析基础》的第四、五、六、七章。分别讲了减治法、分治法、变治法、时空权衡分析。

第四章——减治法：是一种一般性的设计，它利用一个问题给定实例的解和同样问题较小实例的解之间的关系。减治法有3种主要的变化形式：

减一个常量，常常是减1（例如插入排序）

减一个常数因子，常常是减去一个因子2（例如折半查找）

减去可变规模（例如欧几里得算法）

1. 插入排序是减（减一）治技术在排序上的直接问题应用。无论在平均情况还是最差情况下他都是一个θ（n2）的算法，但在平均情况下的效率大约要比最差情况快一倍。
2. 一个有向图是一个对边制定了方向的图。拓扑排序要求按照这种次序列出它的顶点，使得对于中每一条边来说，边的起始顶点总是排在边的结束顶点之前。当且仅当有向图是一个五环有向图（不包含回路的有向图）时，该问题有解，也就是说他不包括有向回路。
3. 解决拓扑排序问题有两种方法。一种算法是基于非常自然的选择。第二种算法是基于减一技术直接应用。
4. 在设计生成基本组合对象的算法时，减一技术是一种非常自然的选择。这类算法中最高效的类型是最小变化算法。然而，组合对象的增长得如此之快，是的实际应用中，即使最高效率也只能用来解决这类问题的一些非常小的实例。
5. 折半查找是一种非常有效的搜索有序数组的算法。他是减常因子算法的一个重要例子。其他例子包括平方求幂、天平选假币、俄式乘法以及约瑟夫斯问题。
6. 对于某些基于减治技术的算法。在算法的一次迭代和另一次迭代时削减的规模是变化的。这种减可变规模算法的例子包括欧几里得、选择问题的基于划分算法、插值查找和二叉查找树中的查找及插入操作。

第五章——分治法，是一种一般性的算法设计技术，他将问题实例分为若干个较小的实例，对这些较小的实例递归求解，然后合并这些解，以得到原始问题的解。分治法是按照以下方案工作的：将一个问题划分为同一类型的若干子问题，子问题最好规模相同；对这些问题求解（一般用递归方法，但在问题规模足够小时，有时也会利用另一个算法）；有必要的话，合并这些问题的解，以得以这些问题的解，以求到原始问题的答案。在这这一章中对合并排序、快速排序、二叉树遍历及其相关特性、大整数和steassen矩阵乘法和用分治法解最对问题和凸包问题的分析。

（1）许多分治算法的时间效率T(n) = aT(n/b) + f(n)。主定理确定了该方程的增长次数。

（2）合并排序是一种分治算法。他把一个数组二分为一，并对它们递归排序，然后把这两个排序好序的子数组合并为原数组的一个有序排序。子啊任何情况下，这个算法的时间效率都是（nlongn），而且它的键值比较次数非常接近理论最小值。它的主要缺点是需要相当大额外存储空间。

（3）快速排序是一种分治排序算法，他根据元素值和某些实现确定的元素比较结果，来对输入元素进行划分它的最差效率是平方级的，平均效率为nlongn

（4）二叉树的经典遍历算法（前序、中序、后序）和其他类似的算法都需要递归处理左右两颗树，他们都可以当做分治技术的例子。用一些外部节点来代替给定的空树，有助于这些算法进行分析。

（5）处理两个n位整数相乘分治算法，大约需要做n1.585次以为数乘法。

（6）Strassen算法值需要做7次乘法就能计算出2\*2矩阵的积，但比基于定义的算法做更多的假发。利用分治技术，该算法计算两个n\*n矩阵乘法时需要做n2.807次乘法。

（7）分治算法还成功的应用于两个重要的计算集合问题：最近对问题和凸包问题。

第六章——变治法。基于变换的思想。因为这些方法分成两个阶段工作的。首先，在“变”的阶段，处于这样或者那样的原因，把问题的实例变得更容易求解。然后，在第二阶段 或者说“治”的阶段，对实例进行求解。

变治技术有三种主要的类型：实例花间、改变表现和问题化简。

1. 实例化简是一种吧问题的实例变换成相同问题的另一个实例技术，这个新的实例有一些特殊的属性，使得它更容易被解决。列表与排序、搞事消去法和AVL树是这种技术的好例子。
2. 改变表现指的是将一个问题实例的表现改变为同样实例的另一种表现。本章所讨论的例子有用2-3树表示集合、堆和堆排序、求多项式的霍纳法则以及两种二进制幂算法。
3. 问题化简提上吧一个给定的问题变换为另一个可以用已知算法求解的问题，化简线性规划问题和化简为图题是尤其重要的。
4. 一些用来阐述变治技术的例子恰好是非常重要的数据结构和算法。他们是：堆和堆排序、AVL树和2-3树、高斯消去法以及霍纳法则。
5. 堆是一颗基本完备二叉树，它的键值都是满足父母优势要求。虽然定义为二叉树，但一般用数组来实现堆。堆对于优先队列的高效实现来说尤为重要，同时，堆还是堆排序的基础。
6. 堆排序在理论上是一种重要的排序算法，它的基本思路是，在排序好堆的数组元素后，再从剩余的堆中连续删除最大的元素。无论在最差情况下还是在平均情况下，该算法的运行时间都属于θ(nlongn)，而且他还是在位的排序算法。
7. AVL树是一种在二维树可能到达的广度上尽量平衡的二叉树。平衡是由四种称为旋转的变换来维持。AVL树上的所有基本曹做都属于O(longn)，它消除了经典二叉查找在最差效率上的弊端。
8. 2-3树是一种达到完美平衡的查找树，它允许一个节点最多包含两个键和三个子女。这个思想推而广之，会产生一种非常张瑶的B树。
9. 高斯消去法是一种解线性方程组的算法，他是线性代数中的一种基本算法。它通过把方程组变换成一个具有三角形矩阵的方程来解题，这种方程组很容易用反向替换法求解。高斯消去法大约需要n3/3次乘法运算。
10. 在无需对系数进行预处理的多项式求解算法中，霍纳法则是最优的。它只需要n次乘法和n次假发。它还有一些有用的副产品，例如综合除法算法。
11. 两种二进制幂算法。它们都使用了指数n的二进制表示，但他们按照相反的方向对其进行处理：从左到右和从右到左。
12. 线性规划关心的是最优化一个包含若干变量的线性函数，这个函数受到一些形式为线性不等式的约束。有一些高效的算法可以对这个问题的炮打实例求解，他们包含了成千上万的变量和约束，但不能要求变量必须是整数。如果变量一定要是整数，我们称之为整数线性规划问题。

第七章——时空权衡。作为一种算法设计技术。空间转换时间要比时间转换空间普遍得多。在算法设计中，空间转换时间技术有两种主要类型，输入增强是其中的一种。它的想是对问题输入的部分或者全部做预处理，然后将获得的额外信息进行存储，以加速后面问题的解决。用分布计数进行排序以及一些重要的字符串匹配算法都是基于这个计数的算法。

（1） 分布计数是一种特殊的方法。用来对元素取值来自于一个小集合的列表排序。

（2）用于字符匹配的Horspool算法可以看做Boyer-Moore算法的一个简化版本。两个算法都以输入增强思想为基础，并且从右向左比较模式中的字符。两个算法都是用同样的坏字符号移动表。Boyer-Moore算法还可以使用第二个表，称为好后缀移动表。

（3）第二种使用了空间换时间权衡思想技术称为预构造，它使用额外的空间来实现更快和更方便的数据存取。散列和B树是预构造的重要例子。

（4）散列是一种非常高效的实现字典的方法。它的基本思想是把键映射到一张一维表中，这种表在大小的限制使得他必须采用一种碰撞解决机制，散列的主要类型是开散列（又称分离链，键存储在散列表以外的链表中）以及闭散列（又称为开式寻址，键存储在散列表中）。平均情况下，这两种算法的查找、插入和删除操作的效率都是属于θ(1)的。

（5）B树是一颗平衡查找树，他爸2-3树的思想推广到允许多个键位于同一节点上，它的主要应用是维护存储在磁盘上的数据的类引索信息。通过寻址适当的树的次数，即使对于非常大的文件，我们所实现的查找，插入和删除操作也只需要指向很少几次磁盘存取。

以上是这一周学习的知识。很多知识也是大概过一遍，只知道一些定义和应用，具体的实现过程没有很好的去了解。

**2016-11-6**

**系统控制与信息处理重点实验室**